Utilisation du shell Unix
5 - Processus

Rappel : vous devez simplement effectuer le travail demandé et noter les commandes utilisées et les réponses aux questions dans l'espace laissé libre. **Attention :** l'objectif est que vous soyez capables de refaire SEUL les manipulations décrites.

Pré-requis

Vous devez avoir parcouru les chapitres « Le langage de commande » et « Les processus » dans les diapos de cours : R1.04-IntroSysteme/Unix.pdf

1 Rôle du `PATH`

La variable `PATH` contient la liste des répertoires où le shell va chercher les commandes à exécuter.

1. Donnez la commande qui affiche la valeur de votre PATH.
2. Copiez le fichier `/users/but/info/Public/bravo.c` dans votre répertoire de TP puis compilez-le et lancez-le (3 commandes).
3. Annulez la valeur de la variable `PATH`, puis lancez la commande `ls`. Expliquez ce qui se passe.
4. Donnez la commande qui permet de lancer quand même ls.
5. Définissez `PATH` avec les deux répertoires `/bin` et `/usr/local/bin`. Vérifiez que vous arrivez à lancer `ls` en tapant simplement `'ls'`.
6. Lancez le programme `bravo`. Que se passe-t-il ? Expliquer.
7. Vous pouvez toujours lancer `bravo` en écrivant simplement `./bravo`, mais on préfère souvent ajouter le répertoire courant au `PATH`. Ajoutez le répertoire courant au `PATH` et vérifiez que vous arrivez à lancer `bravo` en tapant simplement `bravo`.

Notez que vous retrouvez le `PATH` d'origine en ouvrant une nouvelle fenêtre terminal (raccourci Ctrl-Shift-N).

2 Processus

Un processus Unix est toujours créé par un autre processus, appelé son *père*.

8. Lancez en **arrière-plan** (faire suivre la commande d'un `&`) un éditeur de texte (`gedit` ou `geany`).

9. Affichez la liste des fils du processus shell courant avec la commande `ps -f`. Comment vérifie-t-on que `bash` est bien le père du processus que vous venez de lancer (`gedit` ou `geany`) ?

10. Qui est le père de la commande `ps` elle-même ?

11. Donnez les commandes qui permettent de vérifier que la variable `$$` contient bien le numéro du processus shell courant.

12. Affichez la liste de tous les processus avec `ps -ef`, observez-la en créant un tube sur `less`.

13. Affichez la liste de tous les processus avec `ps -ef` et filtrez cette liste avec `grep` pour ne retenir que les processus qui vous concernent.

14. Essayez la commande `ps -fU` *`votreLogin`* qui n'affiche que les processus de l'utilisateur indiqué (*votreLogin*). Filtrez le résultat avec `grep` pour ne retenir que les processus `bash`. Qui est le père des processus `bash` ?

15. La commande `ps -ef --forest` affiche la liste de tous les processus sous forme d'arbre. Avec cette commande et `less`, déterminez le chemin complet depuis `bash` jusqu'à la racine de l'arbre (ne donnez que les noms des commandes, sans le répertoire ni les options).

16. Quel est le nom et le numéro du processus à la racine de l'arbre ?

3 Processus et signaux

La commande `kill -`*`signal num`*`...` envoie un *signal* aux processus dont le numéro est donné. Le signal le plus utilisé est `INT` qui interrompt le processus. Ainsi l'appui sur Ctrl-C pendant l'exécution d'un programme demande au shell d'envoyer un signal `INT` au processus en cours.

Récupérez le programme `/users/but/info/Public/boucle.c` et modifiez-le pour qu'il affiche indéfiniment des carreaux. Recompilez-le puis lancez son exécution et arrêtez-le avec Ctrl-C.

17. Lancez `boucle` en *arrière-plan* et tentez de l'arrêter avec Ctrl-C. Que se passe-t-il ? Expliquez.

18. Ouvrez un nouveau terminal (si nécessaire) puis avec `ps` déterminez le numéro du processus `boucle` et le numéro de son père.

19. Envoyez un signal `INT` au processus `boucle` avec `kill`. Que se passe-t-il ?. Expliquez.

20. Envoyez maintenant un signal `INT` au père de `boucle` (le shell qui l'a lancé). Que se passe-t-il ?

    Vous pouvez vérifiez l'effet d'un signal `INT` sur un shell en tapant Ctrl-C dans un terminal.

    Note : le shell qui s'exécute dans une fenêtre `gnome-terminal` lit les commandes sur son entrée standard (le clavier par défaut). Si l'entrée se termine, le shell se termine et la commande se ferme. On peut donc fermer un terminal en tapant Ctrl-D (marque de fin de fichier). Essayez sur un de vos terminaux ouverts.

21. Envoyez maintenant le signal `KILL` au shell. Que se passe-t-il ?

    Contrairement au signal `INT`, le signal `KILL` ne peut pas être intercepté par le processus et il force l'arrêt brutal du programme. À ce titre, il ne doit pas être utilisé pour arrêter un programme qui peut l'être par d'autres moyens. Ainsi, si vous *tuez* un processus `geany` ou `gvim`, les modifications qui n'ont pas été enregistrées par l'éditeur seront perdues.

22. On peut lancer un processus en arrière-plan en faisant suivre la commande du caractère `&`. La commande `xterm` ouvre un terminal simple. Donnez la commande lançant en une fois 4 processus `xterm`.

23. Déterminez avec `ps` les numéros des 4 processus `xterm` lancés en arrière plan puis donnez la commande qui envoie un signal `INT` aux quatre processus.

4 Valeur de retour d'un processus

Tout processus Unix retourne à son père une valeur correspondant au résultat de son exécution :

— 0 si tout s'est bien passé ;

— $\neq 0$ s'il y a eu une erreur.

Nous verrons dans un prochain TP comment utiliser cette valeur mais nous pouvons déjà l'observer en utilisant la variable `$?` du shell.

24. Lancez `ls` puis faite afficher `$?`. Quelle est sa valeur ?

25. Même question avec `ls trucmachin`.

26. Récupérez le script shell `/users/but/info/Public/valeurRetour`. Ouvrez-le pour observer son contenu puis testez les 4 cas possibles et vérifiez en affichant `$?`.

Compte-rendu

Vous rendrez à l'enseignant un exemplaire du sujet complété à la fin du TP. Si vous n'avez pas fini, utilisez l'exemplaire qui vous reste pour terminer pendant votre temps libre.